Multirepresentational Architectures for Human-Level Intelligence:
Papers from the AAAI Fall Symposium (FS-09-05)

Representing Problems (and Plans) Using Imagery

Samuel Wintermute
University of Michigan
2260 Hayward St.
Ann Arbor, MI 48109-2121
swinterm@umich.edu

Abstract

In many spatial problems, it can be difficult to create a state representation that is abstract enough so that irrelevant details are ignored, but also accurate enough so that important states of the problem can be differentiated. This is especially difficult for agents that address a variety of problems. A potential way to resolve this difficulty is by using two representations of the spatial state of the problem: one abstract and one concrete, along with internal (imagery) operations that modify the concrete representation based on the contents of the abstract representation. In this paper, we argue that such a system can allow plans and policies to be expressed that can better solve a wider class of problems than would otherwise be possible. An example of such a plan is described. The theoretical aspects of what imagery is, how it differs from other techniques, and why it provides a benefit are explored.

Introduction

Many AI systems have been developed that use multiple internal representations of space to improve efficiency and capability. For example, in domains like the blocks world, it is possible to represent spatial data both in abstract qualitative terms, like "A is on B" and quantitatively, in terms of more concrete perceptual data such as pixels or continuous coordinates. When multiple representations are used, internal operations between them are possible. Commonly, the abstract representation is created based on the contents of the concrete representation. Moreover, in some systems, the reverse of this operation is also possible: structures in the concrete representation can be created internally based on the contents of the abstract representation. This capability is called *imagery*.

Here, we are concerned with examining what role imagery can play in a spatial problem-solving agent: an agent that issues actions in the world to solve problems presented to it (at least chiefly) in spatial terms. This is in contrast, for example, to systems that use imagery to make inferences about abstractly-presented spatial information, such as answering "if A is below C and D is right of C, how does A relate to D?" (e.g., Barkowsky, 2002).

Reasons for using multiple representations have been studied in detail. A common argument is that processing over certain types of data is more efficient within specialized representations than in a single uniform representation. If a representation has a specific structure (such as spatiality), inferences that involve properties related to that structure can be very efficient (Shimojima, 1996). For example, it is easier to make inferences related to geometry problems in a representation that explicitly encodes 2D space than in a purely abstract representation (Larkin & Simon, 1987; Lathrop, 2008). In addition, spatial representations implicitly encode background knowledge about space, mitigating the frame problem present in non-spatial representations of action (Huffman & Laird, 1992; Glasgow, 1995; Kurup & Chandrasekaran, 2006).

We wish to build on these arguments (and on our work in Wintermute and Laird, 2009), presenting a case for a different benefit of imagery. We propose that abstract representations and concrete representations provide different benefits to a problem-solving agent in terms of the quality of solutions it is able to achieve, the efficiency with which those solutions can be generated and represented, and the flexibility the agent has to solve multiple problem types. These benefits can be difficult to achieve within a single representation. However, it is our hypothesis that by using multiple representations with imagery, an agent can leverage the positive aspects of each, resulting in a system that is able to make better decisions across more problems than is possible with either representation alone. This imagery benefit is due to representing a problem simultaneously at multiple levels of abstraction, it is distinct from the previously studied benefit of using a spatially structured representation for spatial inferences, and in theory could be extended beyond spatial representations.

A particular form of imagery is considered here: the use of imagery to simulate a future world state based on the current world state, along with information such as a potential action choice. We will call this form of imagery *simulative imagery*. This use is distinct from other ways an agent might use imagery, such as recalling a memory, imagining a situation based on a text description, or using space as an analogy for a non-spatial problem.

The plan for this paper is as follows. First, a simple definition of simulative imagery will be presented. Then, to ground the discussion, an agent using simulative imagery will be described. It is then explained how imagery affects some of the issues inherent in building a good problem

representation. The example is re-examined, and compared to alternative approaches: representing the problem purely in concrete terms or purely in abstract terms. As it turns out, it is possible to represent the problem in purely abstract terms. However, this requires the perception system to perform internal imagery operations, and it is explained why that approach is undesirable in a general-purpose agent.

Simulative Imagery

Many types of AI systems fit the basic pattern on the left of Figure 1: perceptions are mapped to a high-level problem representation, and decisions are made in terms of that representation, resulting in actions. Details of perception are often ignored when discussing these systems and only the representation is addressed, but a perception system is still at least implicitly part of the agent. For example, in classical planning in the blocks world, the representation consists of predicates like **on(A,B)**, and it is implied that, in an embodied agent, some sort of vision system would build those predicates.

Call the direct output of the agent's sensors $\boldsymbol{P}_{\mathrm{l}}$, for low-level perception. This signal is transformed by the perception system to create a higher-level perception signal, called $\boldsymbol{P}_{\mathrm{h}}$. We will call the highest-level part of the agent that receives this signal and decides what to do next the "decision system" (although it could be argued that decisions are made at many levels in the system). This system maintains an internal representation of the problem state, $\boldsymbol{R}$, calculated as a function of $\boldsymbol{P}_{\mathrm{h}}$, possibly taking into account past observations and background knowledge. The decision system also typically uses a high-level representation of actions: it is rare that actions are considered in terms like "set motor voltage to .236", even though that may be the final output of the agent. So, even in a simple system, there are typically distinct high- and low-level action signals, $\boldsymbol{A}_{\mathrm{h}}$ and $\boldsymbol{A}_{\mathrm{l}}$, and a motor system that creates $\boldsymbol{A}_{\mathrm{l}}$ from $\boldsymbol{A}_{\mathrm{h}}$.

An imagery architecture is shown on the right of Figure 1. A box for the imagery system has been introduced. This system maintains its own representation of the problem state, so the overall architecture now has two representations, $\boldsymbol{R}_{\mathrm{i}}$ (in the imagery system) and $\boldsymbol{R}_{\mathrm{d}}$ (in the decision system). The imagery system also provides an additional level of perceptual and action processing. The output of low-level perception is now provided to the imagery system, so it is called $\boldsymbol{P}_{\mathrm{m}}$, for mid-level perception. This is the signal from which $\boldsymbol{R}_{\mathrm{i}}$ is derived. Processing in the imagery system transforms $\boldsymbol{R}_{\mathrm{i}}$ into $\boldsymbol{P}_{\mathrm{h}}$, which is the perception signal provided to the decision system. Note that this happens independently of whether the contents of $\boldsymbol{R}_{\mathrm{i}}$ is real or imagined: the form of $\boldsymbol{P}_{\mathrm{h}}$ is the same, just possibly annotated as real or imagined. That is, the imagery system performs the same high-level perception over both real and imagined data.

The action system is similarly decomposed. Agents can thus be built where the decision system can issue actions that either cause actual action in the world, or simulate the

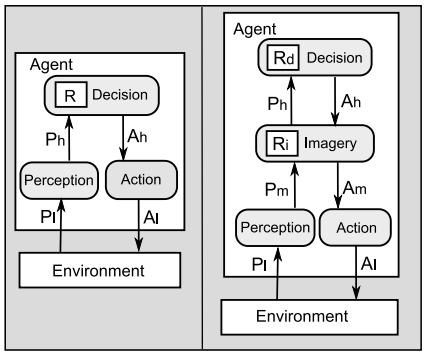

Figure 1: A simple non-imagery AI architecture (left), an imagery architecture (right)

results of that action in the imagery system. These imagery actions allow the agent to predict the value of $\boldsymbol{P}_{\mathbf{h}}$ that a given action would cause if it were to be executed in the environment. Through simulative imagery, the agent can get information about the state of the world not just via $\boldsymbol{P}_{\mathrm{h}}$ directly, but via predictions about *future* values of $\boldsymbol{P}_{\mathrm{h}}$. These predictions can be based on information not present in $\boldsymbol{R}_{\mathrm{d}}$, but present in $\boldsymbol{R}_{\mathrm{i}}$.

Motivating Problem

For a simple example of how simulative imagery can be used to solve a spatial problem, consider a slightly-modified version of a classic blocks world problem. The goal in this problem is to stack four blocks in a particular configuration, **A** on top of **B** on top of **C** on top of **D**. Unlike the standard blocks world, the blocks cannot be placed freely on a table, rather there are two fixed pegs, and each block has a groove down its back that must be aligned to one of the pegs—essentially, there can only be two towers in the world, and their positions are fixed. Blocks can be moved from the top of one tower to the other, however, the blocks vary in size, and the pegs are close enough that blocks may collide, depending on the exact sizes of the other blocks in the towers. Blocks can also be moved out of the way to a storage bin. Assume that moves between the towers are cheap (cost 1) and moves to and from the bin are expensive (cost 20). In addition, collisions are very expensive (cost 1000). So it is in the agent's best interest to solve a problem by moving blocks between the towers, using the bin only if absolutely necessary, and never causing collisions by attempting to move a block where it cannot fit (the same domain was used in Wintermute and Laird, 2009).

This problem can be represented in terms of both an abstract and concrete state. Assume that the agent uses a similar abstract state to what is normally used in the blocks world. The state includes symbols for the important objects in the world (the blocks, bin and pegs). Predicates about these objects are also encoded: **on(X,Y)**, indicating that block **X** is on object **Y** (which could be the base of a peg, the bin, or another block), **clear(X)**, indicating that block **X** can be moved, and **collided(X,Y)**, for when blocks **X** and **Y** have collided. The initial abstract state of the instances

we will consider is **[on(A,peg1) on(B,peg2) on(C,bin) on(D,B)]**, and the goal state is **[on(A,B) on(B,C) on(C,D) on(D,peg2)]**. In addition to the abstract state, a concrete spatial state is present—the exact shapes and positions of the blocks are encoded in terms of continuous numbers.

In the agent, assume that the move actions can be simulated—the agent can use imagery to predict what would happen in the concrete state if a given block were to be moved, and based on that, can extract a description of the (hypothetical) next abstract state.[1] For now, assume that imagery operations have no cost, as they are internal. With this capability, a plan for this problem can be expressed (Figure 2). The agent makes its move choices based on the abstract state, however, instead of acting solely in the world, it can also perform imagery actions. Then, based on the results of imagery, an external action can be chosen. Imagery will sometimes provide differing predictions for states that are identical at the abstract level: for example, all problem instances have the same initial abstract state, but in some instances, moving block **D** to the top of block **B** will cause a collision, and in some instances, it will not. This is represented in the plan as a branch: the next abstract state reached and the associated action chosen differs based on the results of simulative imagery. This happens at several points in the plan, but it is not necessary to use imagery before every move (for instance, moving a block to the bin is always successful, so there is no reason to simulate it).

The plan in the figure shows how the agent acts in four canonical cases where potential collisions arise at different points. In case 1, there is no way to productively move blocks between pegs; instead, they must all go into the bin before building the goal stack. In case 2, this is also true, but the agent cannot determine this until it has already made one movement (of **D** to **A**). In case 3, block **D** never has to be moved to the bin, and in case 4, neither **A** nor **D** ever need to be moved to the bin.

This plan is an example of how the solution to a problem can be *represented* in a system using simulative imagery. This paper does not consider how such a plan could be *generated*. Looking only at how solutions can be represented, the discussion here applies independently of

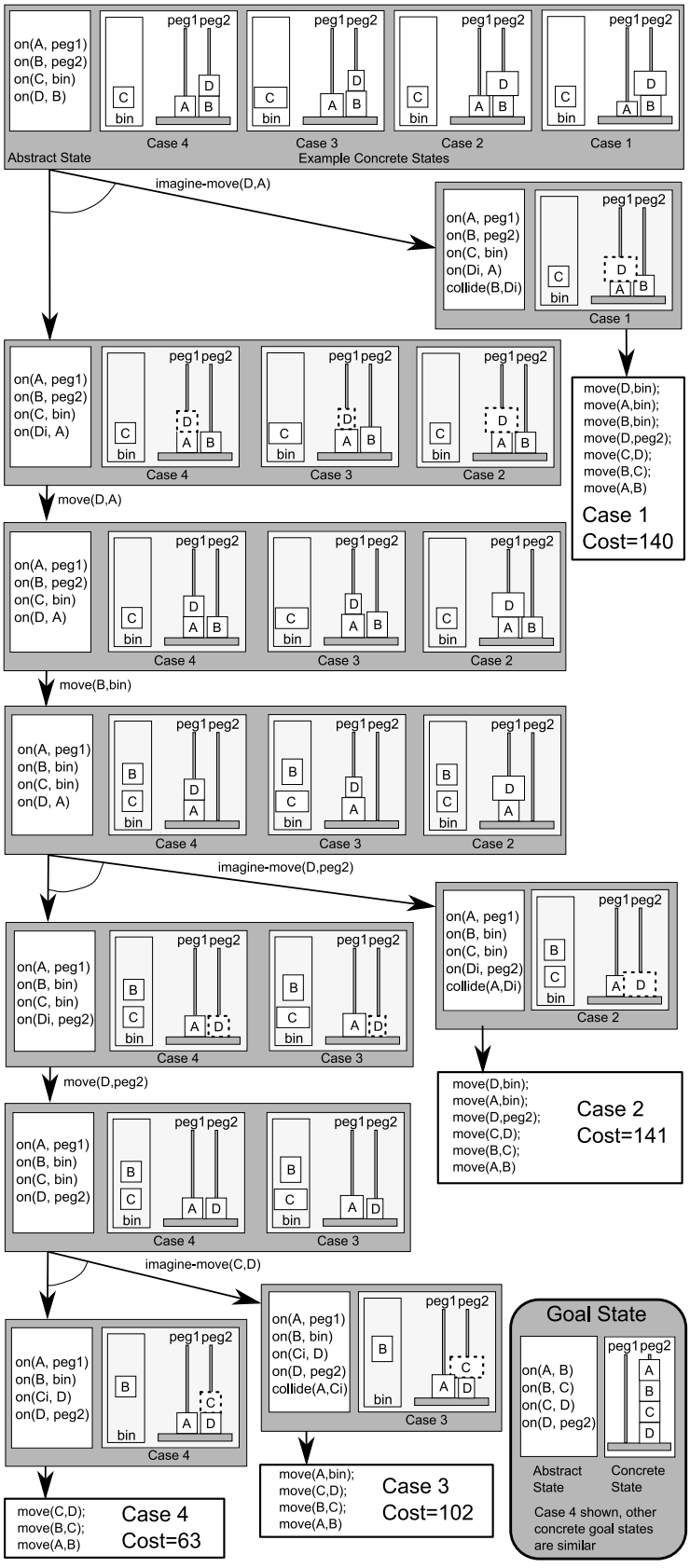

Figure 2 (right): An imagery-conditional plan for pegged blocks world problems. This plan covers instances where the initial abstract state is in the upper left of the figure, and the goal is to stack A on B on C on D on peg2. Four canonical examples of block sizes that lead to four different outcomes are shown. Each path is shown to the point that imagery is no longer needed, after which the actions in the final box are executed in order to reach the goal. Abstract state representations also include "clear" predicates, not shown.

[1] Note that simulative imagery is used here for one-step lookahead, but the same mechanisms could be used for arbitrarily deep search in this domain. One-step lookahead is used to make this agent more representative of agents in domains where aspects of the environment are difficult to model (e.g., the actions of others, or random events), making deep searches unreliable, while the local consequences of the agent's own actions may still be very predictable.

how an agent might be programmed to determine a solution. In particular, this plan can be considered as a subset of an overall *policy* (an association of states to actions), for which the issues involved are the same.

An agent using this plan is implemented in the Soar cognitive architecture (Laird, 2008) using SVS, a module that allows specialized processing for visual and spatial information, including imagery. Detailed descriptions of this agent and the SVS architecture are contained in (Wintermute, 2009). For this discussion, the details of implementation are unimportant, other than to demonstrate that it is possible to represent and execute the plan.

A Tradeoff in Problem State Representation

An imagery system includes two parallel representations of problem state, and the problem is solved through interactions between them. The example above shows how this can be done, but the question remains as to *why* it is an appropriate approach—why is a single representation inadequate? To get at this question, some issues involved in problem representation need to be examined in detail.

For our purposes, a *problem type* can be defined as a criterion on a desired state of the world (a goal), and a *problem instance* as a specific initial state in the world for a given problem. For example, the problem type in Figure 2 has a goal of stacking the blocks in a certain tower, and the figure shows four representative instances of the type.

To connect with existing work, we will consider the problem of creating a unitary state representation of the pegged blocks world task above as a Markov Decision Process (MDP, Sutton & Barto, 1998). An MDP is a set of states, actions, state transition probabilities for each action in each state, and rewards for those transitions. Note that the agent above solves an entire problem type, not just a single instance: this is also desired of the MDP. Consider representing the state here in terms of spatial information: coordinates describing the locations and shapes of the blocks in space. This state information is sufficient such that an optimal action sequence can be determined given an initial state. If there is a finite number of possible blocks, it is possible to induce a complete MDP for the problem with this representation. Call this spatial MDP $\mathbf{M}_s$. Note that $\mathbf{M}_s$ has a very large state space: each instance with the slightest difference in a block dimension is in its own region of that space. To learn or explicitly encode the transitions of this MDP (let alone a policy) would require each possible instance to be separately considered.

However, there is a great similarity across instances of the problem: indeed, as explained above, all instances can be grouped into four cases where the optimal action sequence differs. It is possible to describe a "reduced" version of $\mathbf{M}_s$ (Givan et al., 2003). This MDP, which we will call $\mathbf{M}_r$, represents the same problem as $\mathbf{M}_s$, but with fewer states. Essentially, $\mathbf{M}_r$ is constructed by grouping together states of $\mathbf{M}_s$ that are functionally equivalent[2]. Here, we call the reduced MDP for a particular spatial problem type with the minimal number of states the *ideal* MDP representation of that problem.[3]

While such an ideal representation can be theoretically described, computing it is another matter. Building $\mathbf{M}_r$ automatically by first constructing and explicitly analyzing $\mathbf{M}_s$ would be intractable due to the huge state space of $\mathbf{M}_s$. A more feasible approach often followed is to come up a set of predicates (by hand) that capture the same state distinctions that would be the outcome of an MDP reduction, and design the perception system of the agent to directly provide state information in terms those predicates. For example, in a standard blocks world problem, a set of **on** predicates may be sufficient to capture the state in an ideal representation. A programmer can design a perception system to directly calculate those predicates, rather than relying on an algorithm to automatically infer equivalent states based on a more concrete representation.

Indeed, much work in agent development goes into designing predicates that can create ideal (or close to ideal) representations of spatial problems. Further breaking down what makes an ideal representation can be helpful to understand why this is difficult. Informally, a problem representation can be considered as trading off between two properties: *problem accuracy* and *abstraction*. Problem accuracy corresponds to the degree to which the representation makes all useful distinctions between the states of a problem, and abstraction corresponds to the degree to which the representation maps many states of the world to fewer internal states. An ideal spatial representation is maximal in both of these dimensions: it has all of the accuracy of a concrete spatial representation, but with much more abstraction.

Non-ideal representations are lacking in one (or both) of these properties. A representation low in abstraction is inefficient: if raw spatial information is used, repeated states will rarely be encountered. Minor changes between instances will result in a completely different set of reachable states. With no abstraction, a policy would have to encode actions for each instance separately, or planning rules would have to be written separately for each instance.

Having an abstract representation with fewer states can then lead to faster learning and easier encoding of planning knowledge across instances, since more states of the world will appear similar to the agent. As details are discarded to make a more abstract representation, though, it is difficult to maintain accuracy—it becomes more likely that important problem states will be aliased. This can then lead to a loss in the solution quality possible for the agent to achieve with the representation.

A reasonable aim is to attempt to come up with a representation of space that is more abstract than a

[2] Formally, what is meant here is equivalence under bisimulation (Givan et al., 2003).

[3] Of course, this definition only holds relative to a more primitive (unreduced) MDP. We are looking at spatial problems, so "ideal" in this context means with respect to a concrete spatial MDP of the same problem (that is, an MDP with states encoding the exact polyhedrons involved in 3d space).

concrete representation, but still accurate for all problems the agent must solve. However, the poverty conjecture of Forbus et al. (1991), that "there is no purely qualitative, general-purpose, representation of spatial properties"—can be interpreted as saying that there is no representation for spatial problems that is highly abstract, accurate, and problem-independent. This conjecture, if true, implies that there is no ideal representation that covers all problem domains: a cause for concern if we are designing the perception system of what is intended to be a general-purpose AI system. Any abstract representation of spatial properties that might be built by such a perception system will be inaccurate in at least some problem types.

In many classical AI problems, the world is constrained such that an ideal representation is possible, and systems are built to address a single problem type, or a small family of problem types, so generality is not a concern. In the standard blocks world, **on** predicates concisely capture all important aspects of the world and the problem can be solved with planning solely in terms of those predicates. A state representation based on those same predicates will not work as well (will have less accuracy) in the pegged blocks world, and will have very little accuracy for other spatial problems, like pathfinding, where those predicates do not capture important states of the problem.

Classical AI systems usually cannot deal well with inaccurate state representations. More modern AI systems can model inaccuracy as nondeterminism. If there are two states that appear the same at the abstract level, but in which a given action might lead to two different successor states, that transition can be considered probabilistic. However, this approach is still problematic. There are many cases where the environment and the nature of the agent's sensors or effectors do not allow the problem to be considered deterministic, but those are distinct from the case where the agent's own perception system introduces that aspect to the problem as it builds an abstract representation. In those cases, we may be able to do better, since the information needed to properly differentiate states is present within the agent (in $\boldsymbol{P}_{\mathrm{l}}$), just not in $\boldsymbol{P}_{\mathrm{h}}$.

Imagery provides the capability for an agent to use multiple parallel representations, which differ from each other in terms of abstraction and accuracy. In particular, an imagery system has a decision-level representation that is highly abstract (and possibly inaccurate), and an imagery representation that is not very abstract, but highly accurate. Decisions are made in terms of the abstract representation, aided by predictions made in terms of the imagery representation. The central hypothesis of this work is that, taken as a whole, the behavior of such a system can be reflect the best of both of these representations, while maintaining generality across problems.

Revisiting the Example

These principles can be clarified by taking another look at the example plan above. Consider the problem of designing a system to achieve the same performance on the same problem instances, but without using imagery.

One option is to use only an abstract representation, for example, representing each state as simply a set of **on** and **clear** predicates. A plan to stack the blocks without collision in all instances can be expressed: always move all of the blocks to the bin, and restack them into the goal configuration. This treats every instance as case 1. This will work, but this representation clearly lacks accuracy: it does not distinguish between states well enough to separate the cases where less expensive solutions are possible (specifically, cases 3 and 4). Across all problem instances, the agent will not perform as well as the imagery agent.

A similar naïve approach is to not bother with an abstract representation, and simply use concrete spatial information as the problem state. For any given problem instance, a collision-free solution plan can be expressed. However, each plan applies reliably only in that exact same instance: a minor change to a block's size can change the solution to the problem. To cover all of the instances the imagery agent is able to cover requires a huge number of plans. Again, this is inferior to using imagery.

Perhaps the problem here is that the above abstract representation is too simplistic. Maybe an abstract representation is still possible, but more predicates are needed, such as **wider-than(X,Y)** and **taller-than(X,Y)**. This is very difficult, though: consider differentiating between cases 3 and 4. To determine whether or not **C** will collide with **A**, the width of **C** is important, along with both the width and height of **A**, the height of a third block, **D**, and the distance between the two pegs. If determining an abstract representation of this still is not hard enough, the shapes of the "blocks" could be changed in arbitrary ways (perhaps by giving them puzzle-piece edges). The imagery agent can take these manipulations in stride – its behavior is conditional on whether or not things intersect, a property that is easy to compute based on the imagery representation, but is extremely difficult to capture in an abstract representation.

Imagery vs. More Complex Perception

However complex it may be, it is still possible to create a problem-specific abstract representation of the pegged blocks world problem without sacrificing accuracy. Consider if the perception system in the agent provided problem-specific predicates that exactly distinguish between the four cases, like **collision-if-X-on-peg1-and-Y-on-Z-on-peg2(X,Y,Z)**.

Such an abstract representation can capture the same aspects of the problem as the imagery-conditional plan, and could achieve the same performance on the same problem instances.[4] Using imagery requires complicating the architecture, adding in extra modules and connections, whereas the alternative just requires a smarter perception system. Why then should we bother with the architectural overhead of imagery, when good perception results in the same performance?

[4] It could achieve slightly better performance if cases 1 and 2 could be distinguished before taking any actions.

Vision, as it is usually considered, is a process of combining local information from sensors, such as edges and textures, and building up to progressively higher-level representations, such as objects and relationships between them. This is a process that likely must have both parallel, bottom-up aspects, and serial, top-down aspects (Ullman, 1984). However, there is an important question as to whether those top-down aspects should be under the direct control of the decision-making part of the agent. If simple top-down operations (such as tracing the boundary of a region) are needed for task-independent object recognition, there is not likely to be a huge need for this level of control: the agent would always choose to recognize the objects around it. In our problem, to extract the complex collision predicate above, a perception system could internally use a spatial representation, and explicitly copy the block to the peg location and check for intersections, just as imagery does. Should this procedure then also be automatic and hidden from the decision-making part of the agent, just like those procedures that might be used to recognize the block?

When designing an architecture to cover many problems, this is not an appropriate approach. The needed result of perception here is very problem-specific. If the agent's task is not to stack blocks on pegs, but instead is to arrange them in the bin by color, the complex collision predicate introduced above would be useless. Imagery allows the same information to be extracted from the perception system through a sequence of simple operations over time, rather than relying on complicated problem-specific perception processes. This means that plans and policies for very different problems can be represented within the same system, without requiring changes to the lower-level parts of the system. For example, agents have been implemented in Soar/SVS both to cover the problem here and very different problems, such as nonholonomic car motion planning (Wintermute, 2009). These agents differ only in their abstract decision-level knowledge (and in their action systems). They use the same basic operations for imagery and high-level perception.

There are also reasons to prefer imagery to more complex perception even in single-problem agents, such as the pegged blocks world agent in Figure 2. Imagery allows complex properties of the state to be extracted only when they are needed. In Cases 1 and 2, for example, the information equivalent to **collision-if-X-on-peg1-and-Y-on-Z-on-peg2(X,Y,Z)** that separates cases 3 and 4 is never extracted by the agent, as the imagery operations are in a part of the plan that is never reached. And in no case is that information extracted about blocks for which it is not directly relevant. Decomposing complex perception into atomic steps of imagery and simple perception allows the decision system to exercise precise control over it, avoiding unnecessary steps.

Conclusion

In this paper, we have presented an argument for how representing a spatial problem using simulative imagery can allow for better agent behavior. Using imagery, behavior can be dependent both on an abstract description of the world, and on internally-generated predictions about the abstract consequences of actions. This allows plans to be expressed that leverage the benefits of abstract representation, while counteracting problems that can occur when different states are aliased together. An alternative to this approach is to require the perception system to build a problem-specific abstract representation. However, this is not appropriate in an agent that addresses multiple problems, and even for some single-problem agents, there are good reasons to prefer using imagery.

Acknowledgements

John Laird provided guidance on this project, and helped edit the paper. This research was funded by a grant from US Army TARDEC.

References

Barkowsky, T. (2002). *Mental Representation and Processing of Geographic Knowledge: A Computational Approach*. Springer.

Forbus, K. D., Nielsen, P., & Faltings, B. (1991). Qualitative spatial reasoning: the CLOCK project. *Artificial Intelligence*, *51*(1-3), 417-471.

Givan, R., Dean, T., & Greig, M. (2003). Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence, 147*(1) 163–224.

Glasgow, J. (1995). A formalism for model-based spatial planning. In *Spatial Information Theory A Theoretical Basis for GIS* (pp. 501-518).

Huffman, S., & Laird, J. E. (1992). Using Concrete, Perceptually-Based Representations to Avoid the Frame Problem. In *AAAI Spring Symposium on Reasoning with Diagrammatic Representations*.

Kurup, U., & Chandrasekaran, B. (2006). Multi-modal Cognitive Architectures: A Partial Solution to the Frame Problem. In *Proceedings of CogSci 2006.*

Laird, J. E. (2008). Extending the Soar Cognitive Architecture. In *Proceedings of AGI-08.*

Larkin, J.H. & Simon, H.A. (1987). Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11(1), 65-100.

Lathrop, S. D. (2008). *Extending Cognitive Architectures with Spatial and Visual Imagery Mechanisms*. PhD Thesis, University of Michigan.

Shimojima, A. (1996). *On the efficacy of representation*. PhD Thesis, Indiana University.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Ullman, S. (1984). Visual routines. *Cognition*, *18*(1-3), 97.

Wintermute, S., (2009). *An Overview of Spatial Processing in Soar/SVS*. Technical Report, University of Michigan Center for Cognitive Architecture.

Wintermute, S., & Laird, J. E. (2009). Imagery as Compensation for an Imperfect Abstract Problem Representation. In *Proceedings of CogSci 2009.*